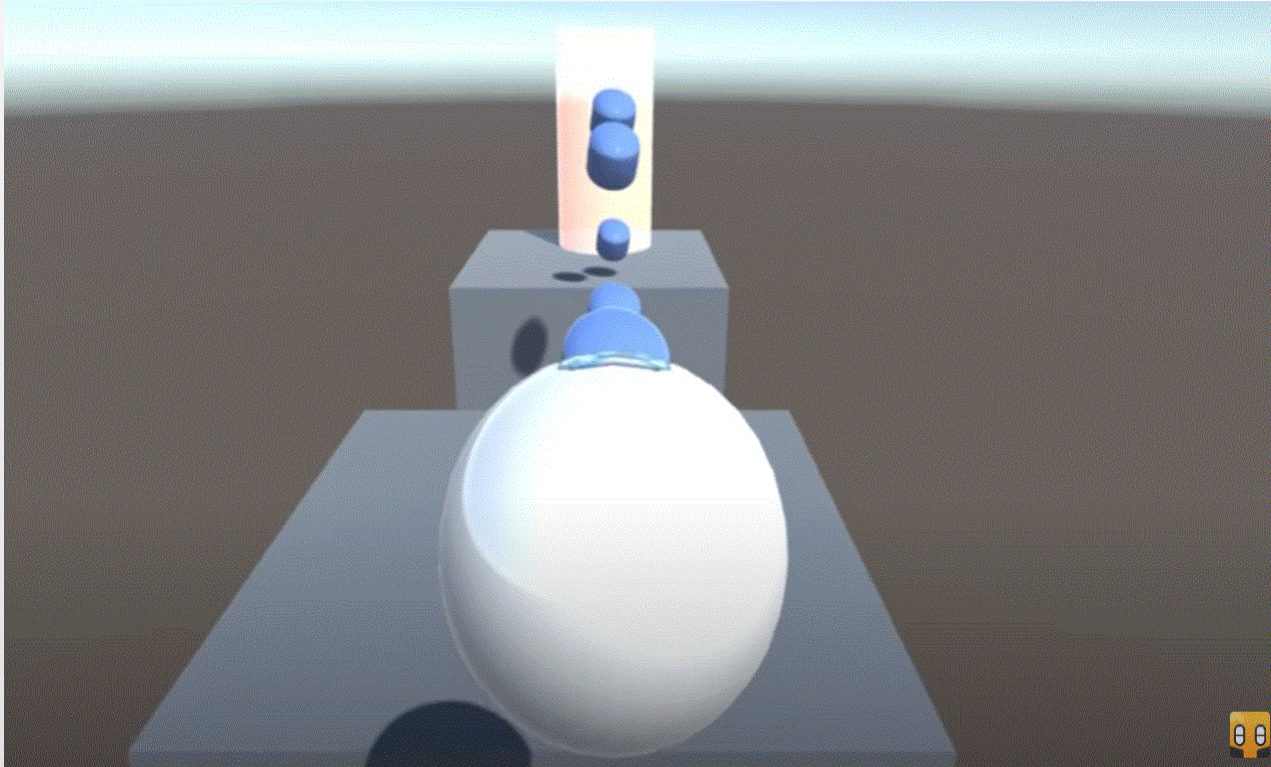


# Unity Project

2018775050 - 이인석

# 게임 설명



- 맵 안에 아이템을 다 먹고 포탈로 들어가는 게임
- 키보드로 플레이어(공)를 조작

## 원본 소스 – Player 변수 및 초기 설정

```
public float jumpPower;  
public int itemCount;  
public GameManagerLogic manager;
```

```
bool isJump;  
Rigidbody rigid;  
AudioSource audio;
```

```
void Awake()  
{  
    isJump = false;  
    rigid = GetComponent<Rigidbody>();  
    audio = GetComponent<AudioSource>();  
}
```

- jumpPower – 점프 강도 변수
- itemCount – 아이템 개수 변수
- Manager – 매니저 객체 정보 변수
- Bool – 점프 상태를 확인하는 변수
- Rigid – rigidbody 컴포넌트 정보를 받는 변수
- Audio – AudioSource 컴포넌트 정보를 받는 변수
- Awake() – Start()보다 먼저 실행합니다.



## 원본 소스 – Player 움직이는 함수

```
void FixedUpdate()
{
    float h = Input.GetAxisRaw("Horizontal");
    float v = Input.GetAxisRaw("Vertical");

    rigid.AddForce(new Vector3(h, 0, v), ForceMode.Impulse);
}
```

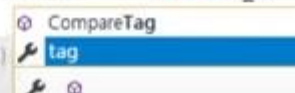
```
void Update()
{
    if (Input.GetButtonDown("Jump") && !isJump) {
        isJump = true;
        rigid.AddForce(new Vector3(0, jumpPower, 0), ForceMode.Impulse);
    }
}
```

- FixedUpdate() - H, V 변수에 방향키를 입력을 즉시 반응하기 위해 GetAxisRaw 함수 사용하고 rigid x와 z 좌표를 이동하기 위해 AddForce() 함수 사용
- Update() – 점프버튼을 누르고 isJump가 false 일때 isJump가 true로 변환하고 jumpPower 변수 만큼 y좌표가 이동
- FixedUpdate() - 프레임을 기반으로 호출되는 Update 와 달리 Fixed Timestep에 설정된 값에 따라 일정한 간격으로 호출됩니다.
- Update() - 스크립트가 enabled 상태일때, 매 프레임마다 호출됩니다.

## 원본 소스 – Player 충돌 함수

```
void OnTriggerEnter(Collider other)
{
    if (other.tag == "Item") {
        itemCount++;
        audio.Play();
        other.gameObject.SetActive(false);
        manager.GetItem(itemCount);
    }
    else if (other.tag == "Point") {
        if(itemCount == manager.totalItemCount) {
            //Game Clear! && Next Stage
            if (manager.stage == 2)
                SceneManager.LoadScene(0);
            else
                SceneManager.LoadScene(manager.stage + 1);
        }
        else {
            //Restart Stage
            SceneManager.LoadScene(manager.stage);
        }
    }
}
```

```
void OnCollisionEnter(Collision collision)
{
    if (collision.gameObject.tag == "Floor")
        isJump = false;
}
```

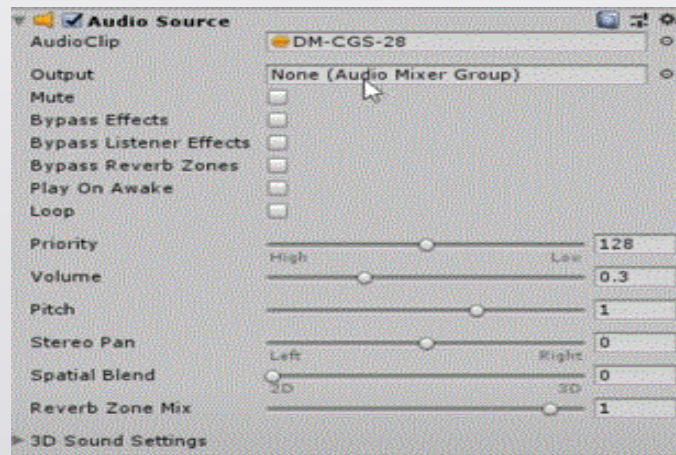


- OnTriggerEnter() – 만약 Item 태그를 가진 객체와 충돌하면 itemCount 변수 상승, 소리 출력, 충돌한 객체는 화면에서 삭제, manager 객체에 itemCount 값을 전달 또는 Point 태그를 가진 객체와 충돌하면 itemCount가 manager 객체의 totalItemcount랑 같으면 스테이지 이동 아니면 현재 스테이지 재 시작
- OnCollisionEnter() – 만약 Floor 태그를 가진 객체와 충돌 시 isJump 변수 false로 변경

## 원본 소스 – Item 회전함수 및 Audio 컴포넌트 및 태그 설정

```
public class ItemCan : MonoBehaviour
{
    public float rotateSpeed;

    void Update()
    {
        transform.Rotate(Vector3.up * rotateSpeed * Time.deltaTime, Space.World)
    }
}
```



Tag 0	Item
Tag 1	Floor
Tag 2	(Removed)
Tag 3	Point

- rotateSpeed –회전 속도 변수
- Rotate() – Rotate() 함수로 초당 rotateSpeed 변수값만큼 월드 기준으로 회전
- Audio - 다운로드한 Audio 소리 파일을 AudioClip에 적용
- Tag - 각각의 객체에 사용할 태그 추가



# 원본 소스 – GameManagerLogic 및 GameManager 객체 컴포넌트

```
public class GameManagerLogic : MonoBehaviour
{
    public int totalItemCount;
    public int stage;
    public Text stageCountText;
    public Text playerCountText;

    void Awake()
    {
        stageCountText.text = "/" + totalItemCount;
    }

    public void GetItem(int count)
    {
        playerCountText.text = count.ToString();
    }

    void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.tag == "Player")
            SceneManager.LoadScene(stage);
    }
}
```



- totalItemCount – 아이템 먹은 값 변수
- Stage – 스테이지 변수
- stageCountText – UI에 스테이지값을 나타내는 변수
- playerCountText – UI에 아이템 먹은 값을 나타내는 변수
- Awake() - stageCountText와 playerCountText 초기화
- Getitem() – 아이템을 먹을때 마다 playerCountText가 count값으로 바뀜
- OnTriggerEnter() – 만약 Palyer태그를 가진 오브젝트가 충돌하면 스테이지 씬으로 이동

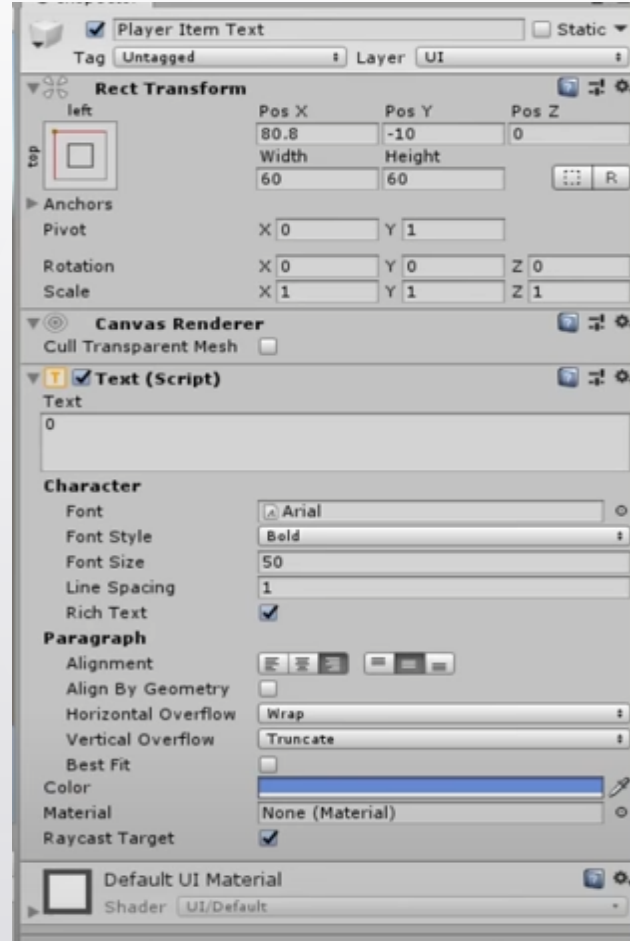
# 원본 소스 - 카메라 위치 코드

```
Transform playerTransform;  
Vector3 Offset;  
  
void Awake()  
{  
    playerTransform = GameObject.FindGameObjectWithTag("Player").transform;  
    Offset = transform.position - playerTransform.position;  
}  
  
void LateUpdate()  
{  
    transform.position = playerTransform.position + Offset;  
}
```

- Awake() - 시작하기 전에 Player 태그를 가진 오브젝트의 트랜스폼의 정보 변수 선언과 카메라 포지션, 플레이어 오브젝트의 포지션 뺀 변수를 선언
- LateUpdate() - 모든 업데이트가 호출이 끝났을 때 카메라의 위치는 플레이어 포지션, offset 변수값의포지션과 합쳐서 오브젝트를 따라가게 설정
- LateUpdate() - 모든 Update 함수가 호출된 후, 마지막으로 호출됩니다.
- Update() - 스크립트가 enabled 상태일때, 매 프레임마다 호출됩니다.

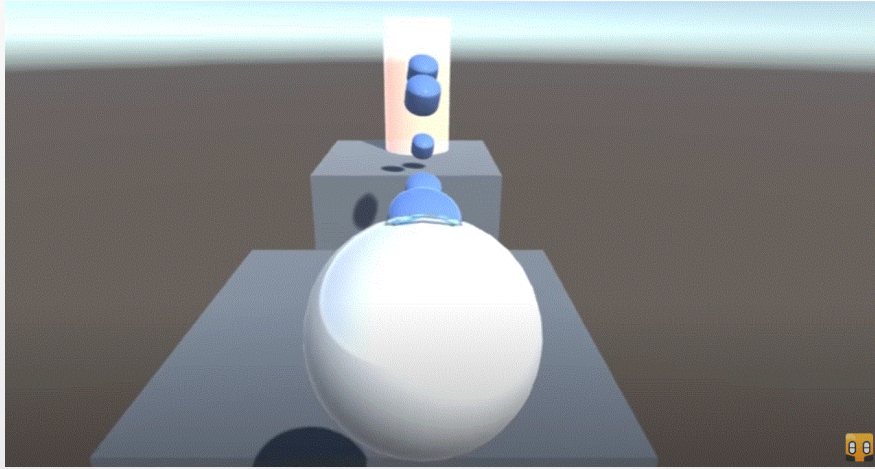


# 원본 소스 - UI 캔버스



- UI 캔버스 추가
- 추가된 UI 캔버스에 현 스테이지를 나타내는 텍스트를 추가
- 추가된 UI 캔버스에 현재 먹은 아이템의 수를 나타내는 텍스트를 추가

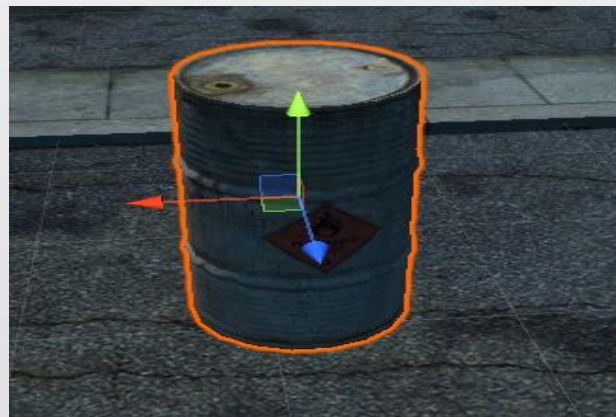
# 게임 수정 및 업그레이드



- Player 오브젝트 변경
- 바닥 오브젝트 변경
- 썬 이동 삭제
- 건물 추가
- 맵 탈출 불가능 벽 추가
- 기름통 오브젝트 추가
- 플레이어 시점 변경, 전체 화면 시점 추가
- 게임 시작 후 서서히 기름 소모 기능 추가
- 기름 없으면 플레이어 작동 불가 기능 추가
- 조작을 안 하면 빠른 속도 감속 기능 추가
- 플레이어 동작 코드 변경
- 버튼으로 플레이어 색 변경 기능 추가
- 게임 시작 시 시동 소리, 운전 중 소리 출력, 기름이 없으면 운전 소리 종료 기능 추가
- 기름통 아이템을 먹을 시 기름양 추가
- 캔버스에 기름양, 스피드 텍스트 추가



# 게임 오브젝트



- 벽 추가
- 자동차 오브젝트 변경
- 기름통 오브젝트 추가
- 건물 추가
- 도로 오브젝트 추가



# Player(차) 바퀴 동작 구현(1)

```
public float power = 50f; // 바퀴를 회전시킬 힘
public float rot = 15f; // 바퀴의 회전 각도
Rigidbody rb;
public WheelCollider[] wheels = new WheelCollider[4];
// 차량 모델의 바퀴 부분 4개
GameObject[] wheelMesh = new GameObject[4];
```

```
public int currentSpeed; // 현재 속도
```

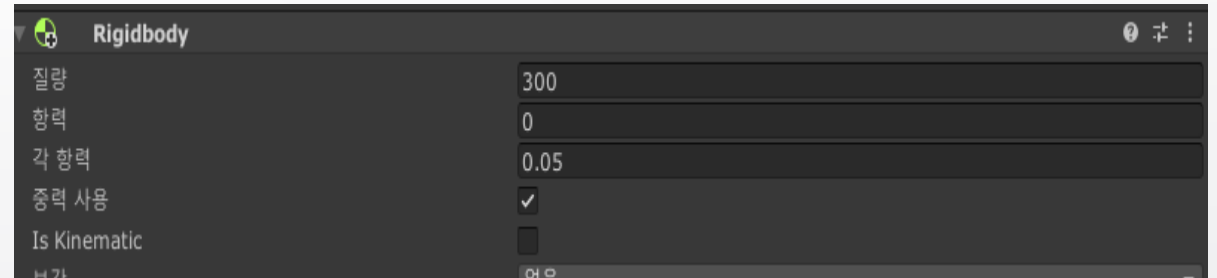
```
public float decSpeed = 100f; // 감속량
```

```
public int Maxoil = 2000; // 기름양
```

```
void Start()
```

```
{
    // 바퀴 모델을 태그를 통해서 찾아온다.(차량이 변경되더라도 자동으로 찾기위해서)
    wheelMesh = GameObject.FindGameObjectsWithTag("WheelMesh");
    StarAudio.Play();
    RidingAudio.Play();
    RidingAudio.loop=true;
}
```

```
for (int i = 0; i < wheelMesh.Length; i++)
{
    // 휠콜라이더의 위치를 바퀴메쉬의 위치로 각각 이동시킨다.
    wheels[i].transform.position = wheelMesh[i].transform.position;
}
rb = GetComponent<Rigidbody>();
// 무게 중심을 y축 아래방향으로 낮춘다.
rb.centerOfMass = new Vector3(0, -1, 0);
}
```



- 바퀴 회전 힘, 회전 각도 값 변수 추가
- Rigidbody가 설정된 오브젝트 변수 추가
- WheelCollider를 사용할 바퀴 모양 오브젝트들 추가
- 바퀴 겉모습을 보여줄 Mesh 오브젝트들 추가
- 현재 속도 변수 및 감속량 변수 추가
- 시작 시 WheelCollider의 위치를 Mesh 오브젝트 위치로 각각 이동, 차가 안정적이게 하기 위해 무게 중심 추가
- 오브젝트 무게를 300으로 설정

## Player(차) 바퀴 동작 구현(2)

```
//기름이 없으면 플레이어 작동불가
if (Maxoil <= 0)
{
    wheels[i].brakeTorque = decSpeed;
    RidingAudio.loop = false;
    RidingAudio.Stop();
}
else
{
    wheels[i].brakeTorque = 0;
    //게임 시작 후 서서히 기름 소모
    Maxoil -= Maxoil * 1;

    // for문을 통해서 휠콜라이더전체를 Vertical 입력에 따라서 power만큼의 힘으로 움직이게 한다.
    wheels[i].motorTorque = Input.GetAxis("Vertical") * power;
    //속도 감소
    if (!Input.GetButton("Vertical"))
    {
        wheels[i].brakeTorque = decSpeed;
    }
    else
    {
        wheels[i].brakeTorque = 0;
    }
}

for (int i = 0; i < 2; i++)
{
    // 앞바퀴만 각도전환이 되어야하므로 for문을 앞바퀴만 해당되도록 설정한다.
    wheels[i].steerAngle = -Input.GetAxis("Horizontal") * rot;
}
```

- 기름이 0보다 작거나 같아지면 brakeTorque 값을 감속량을 넣고 아니면 brakeTorque 값을 0으로 설정 후 기름이 FixedUpdate() 할 때마다 1씩 감소 motorTorque 값은 방향키 입력에 따라 power 만큼의 힘으로 움직이게 설정 이때 방향 키를 입력 안 하면 감속하게 설정
- 앞바퀴만 가져와서 좌우로 입력한만큼 각도 전환 되게 설정

## Player(차) 색 변경

```
public GameObject body;
```

```
public Material white;
```

```
public Material black;
```

```
public Material blue;
```

```
public Material green;
```

```
public Material gold;
```

```
if (Input.GetKey(KeyCode.Keypad1))
```

```
{
```

```
    body.GetComponent<MeshRenderer>().material=white;
```

```
}else if (Input.GetKey(KeyCode.Keypad2))
```

```
{
```

```
    body.GetComponent<MeshRenderer>().material = black;
```

```
}else if (Input.GetKey(KeyCode.Keypad3))
```

```
{
```

```
    body.GetComponent<MeshRenderer>().material = blue;
```

```
}else if (Input.GetKey(KeyCode.Keypad4))
```

```
{
```

```
    body.GetComponent<MeshRenderer>().material = green;
```

```
}
```

```
else if (Input.GetKey(KeyCode.Keypad5))
```

```
{
```

```
    body.GetComponent<MeshRenderer>().material = gold;
```

```
}
```

- 변경할 바디 오브젝트와 각각의 색상의 메타리얼 변수 설정
- 키패드 1~5를 누르면 Player 색이 흰색, 검은색, 파란색, 초록색, 노란색으로 변경되게 설정



# Player(차) 사운드 및 기름통 충돌 시

```
public AudioSource StarAudio;  
public AudioSource RidingAudio;
```

```
StarAudio.Play();  
RidingAudio.Play();  
RidingAudio.loop=true;
```

```
RidingAudio.loop = false;  
RidingAudio.Stop();
```

```
void OnTriggerEnter(Collider other)  
{  
    if(other.tag == "oil")  
    {  
        Maxoil = Maxoil + 500;  
        other.gameObject.SetActive(false);  
    }  
}
```

- 각각의 사운드 변수 설정
- 게임이 시작 시 각각 오디오 플레이 단 Riding 오디오는 무한으로 출력되게 설정
- 만약 기름이 없으면 Riding 오디오 루프 해체 후 사운드 멈추게 설정
- Oil 태그를 가진 오브젝트와 부딪치면 기름양을 증가시키고 그 오브젝트는 안 보이게 설정

# Player(차) 시점과 전체화면 시점 변경 및 UI Text들 변경

```
public GameObject MainCam;
public GameObject PlayerCam;

Unity 메시지 | 참조 0개
void Start()
{
    PlayerCam.SetActive(false);
}

// Update is called once per frame
Unity 메시지 | 참조 0개
void Update()
{
    if (Input.GetKey(KeyCode.Q))
    {
        MainCam.SetActive(true);
        PlayerCam.SetActive(false);
    } else if (Input.GetKey(KeyCode.W))
    {
        MainCam.SetActive(false);
        PlayerCam.SetActive(true);
    }
}
```

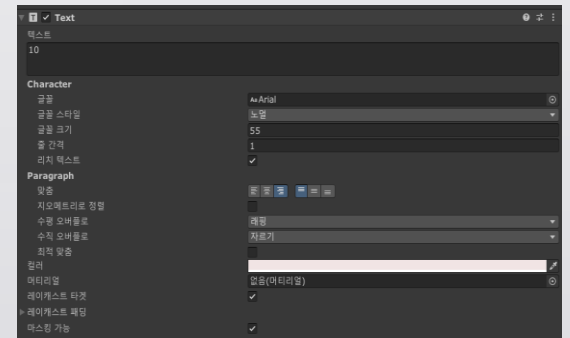
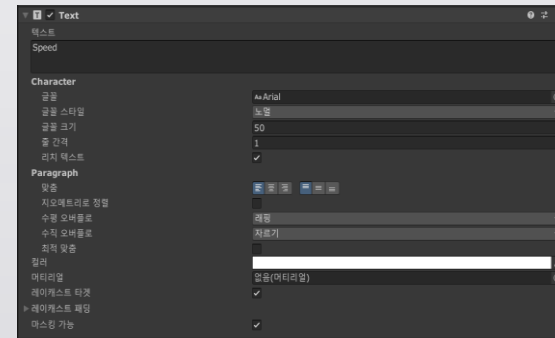
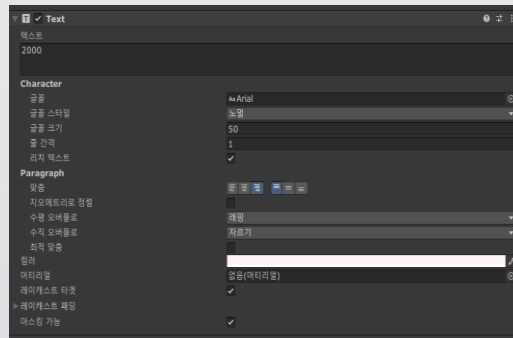
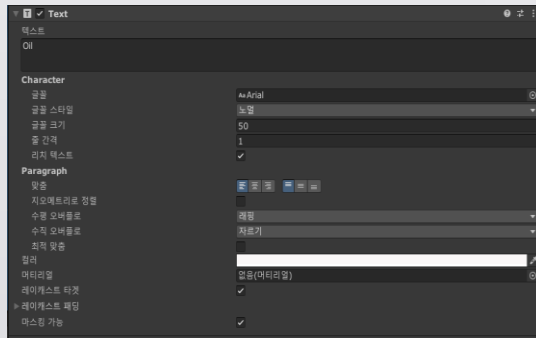
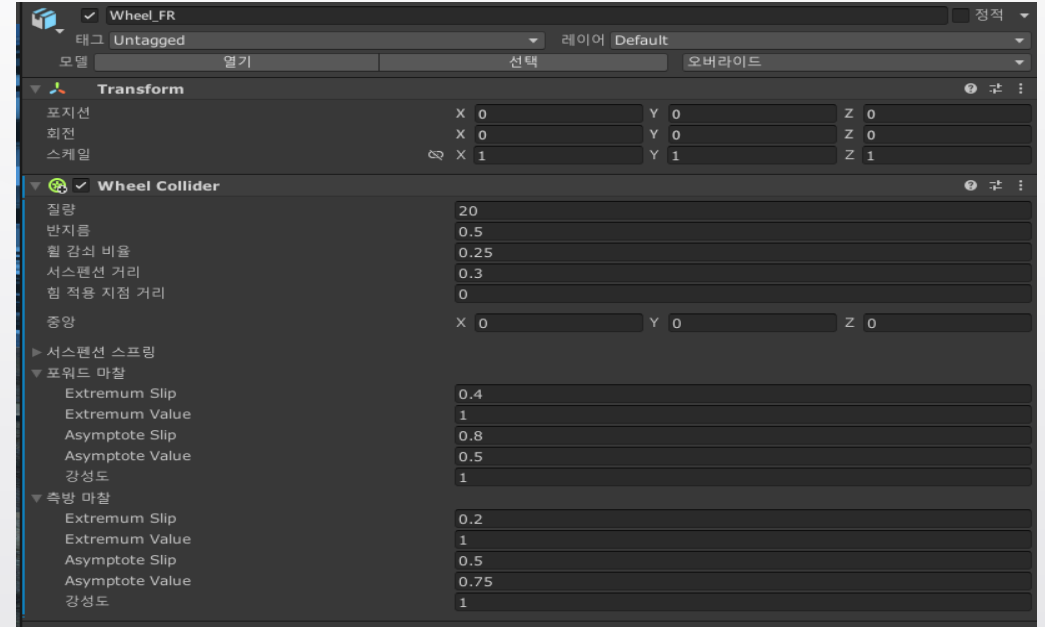
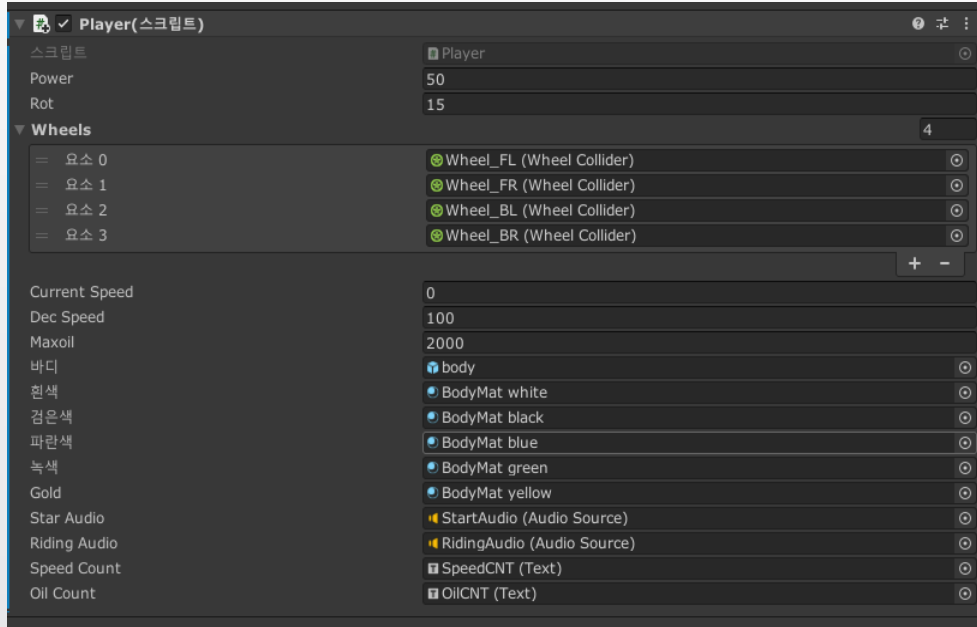
```
public Text SpeedCount;
public Text oilCount;
```

```
public int currentSpeed; // 현재 속도
```

```
currentSpeed = (int)rb.velocity.magnitude;
SpeedCount.text = currentSpeed.ToString();
oilCount.text = Maxoil.ToString();
// 현재 시점 변경
```

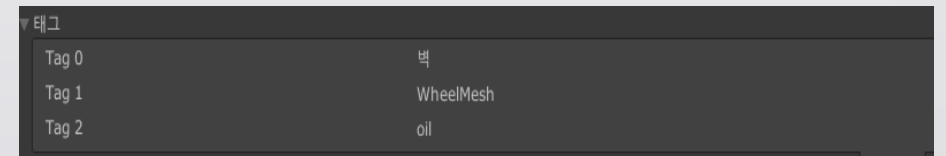
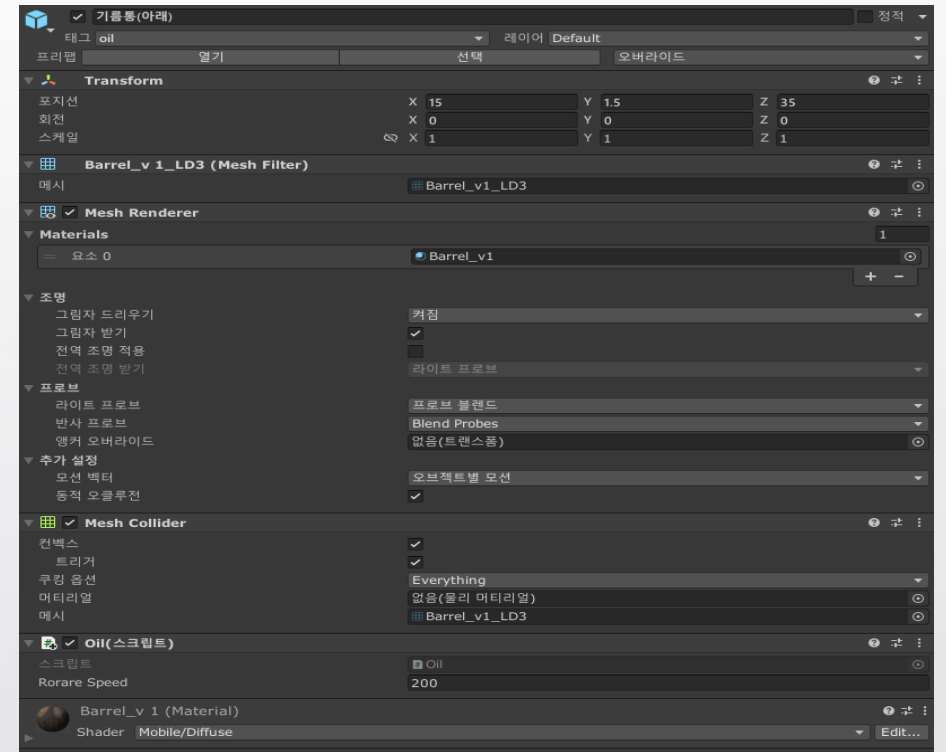
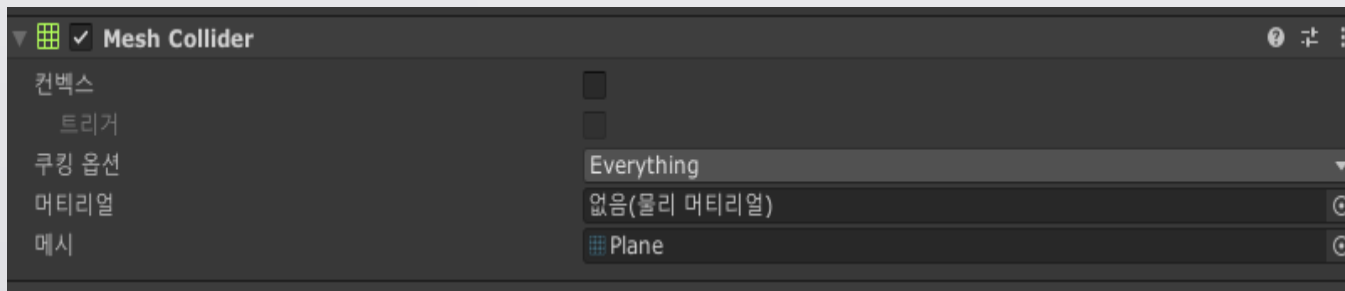
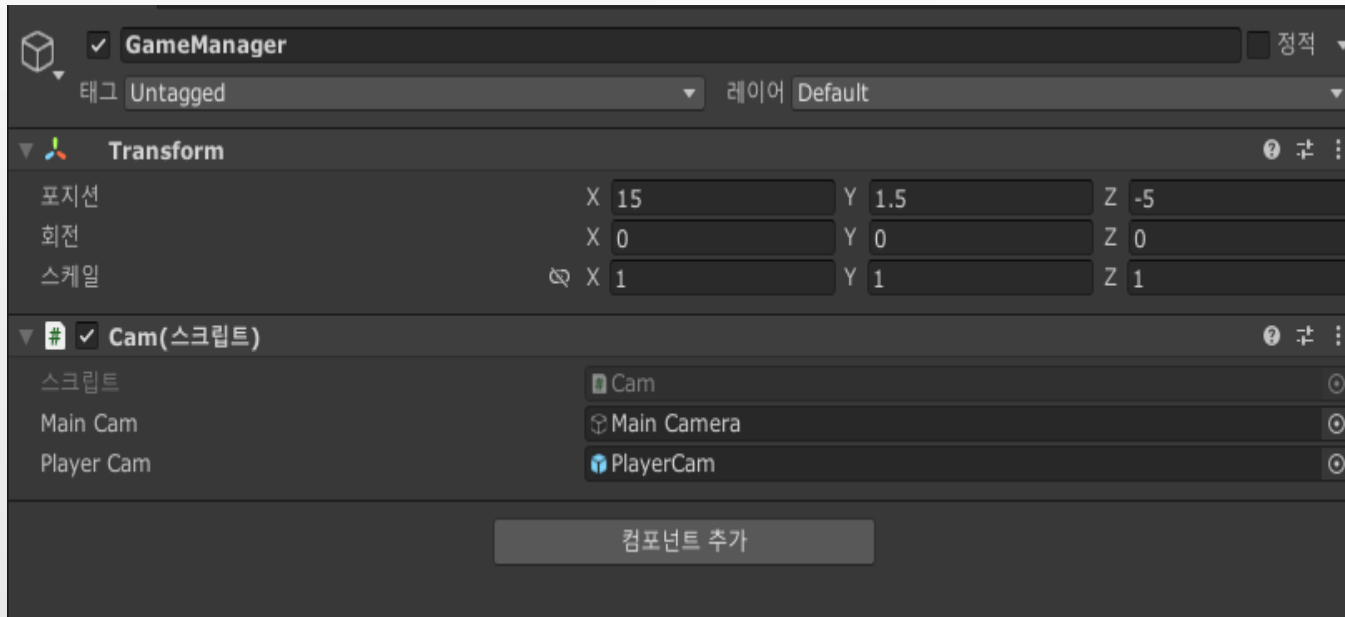
- 시작 시 PlayerCam을 오프 시키고 Q를 누르면 전체 화면 시점 W를 누르면 Player 시점을 전환되게 설정
- UI에서 사용할 Speed Text와 Oil Text들을 현 상태에 따라 Text 변경되게 설정

# Player 스크립트 및 Wheel Collider 설정 및 UI Text들

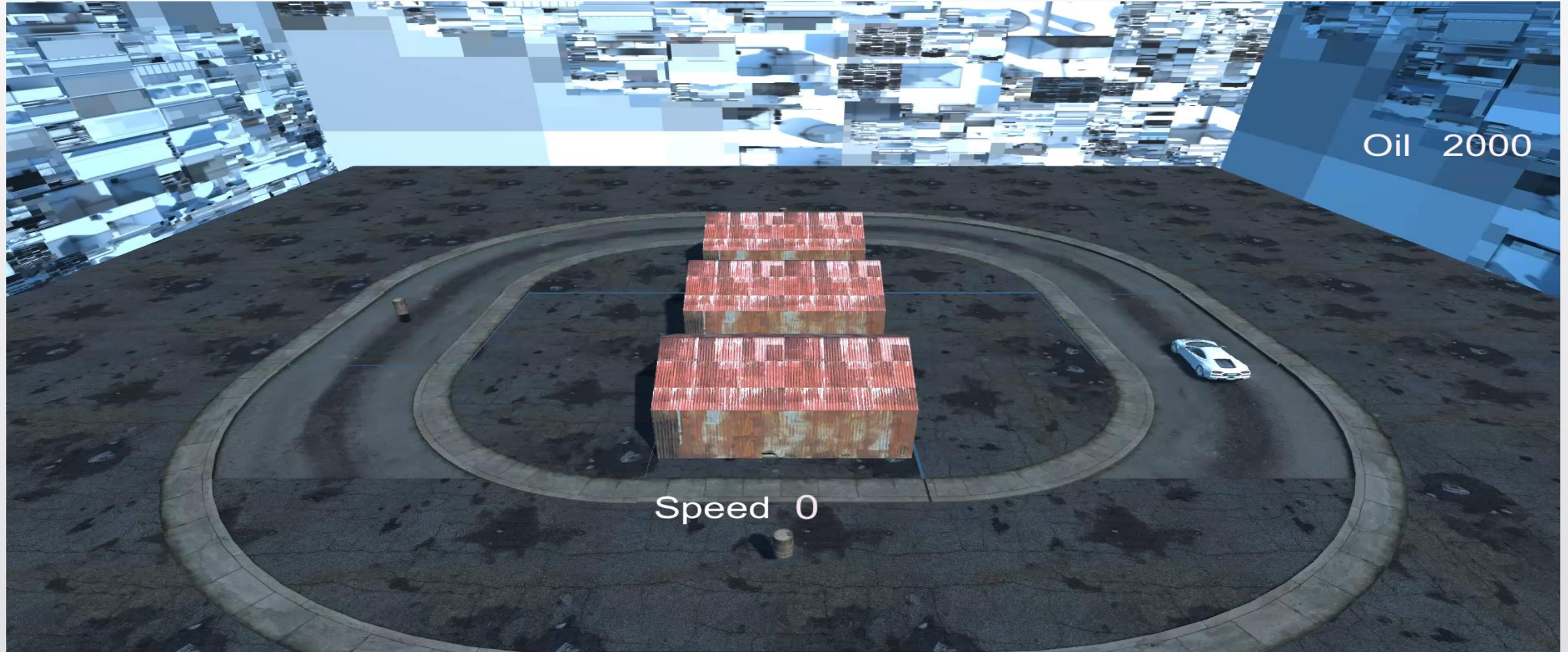




# GameManager 스크립트 및 벽 설정 및 기름통 설정 및 태그 추가



# 게임 영상



# 참고 자료

- 원본 - <https://www.youtube.com/watch?v=pTc1dakebow>
- 휠 클라이더 기본 세팅 - <https://coding-of-today.tistory.com/128>
- 휠 클라이더 움직임 구현 - <https://coding-of-today.tistory.com/130>
- 카메라 온오프 - <https://artiper.tistory.com/106>
- 키 코드 - <https://wergia.tistory.com/211>
- 속도 감속 - <https://micropilot.tistory.com/2656>
- 오디오 - <https://202psj.tistory.com/1312>
- 각각의 업데이트 - <http://developug.blogspot.com/2014/09/update-fixedupdate-lateupdate.html>
- 구조체+맵 - <https://assetstore.unity.com/packages/3d/environments/industrial/rpg-fps-game-assets-for-pc-mobile-industrial-set-v2-0-86679>
- 사운드 - <https://assetstore.unity.com/packages/audio/sound-fx/transportation/i6-german-free-engine-sound-pack-106037>
- 자동차 - <https://assetstore.unity.com/packages/3d/vehicles/land/hq-racing-car-model-no-1203-139221>



감사합니다.